CirCUs: A Satisfiability Solver Geared towards Bounded Model Checking*

HoonSang Jin, Mohammad Awedh, and Fabio Somenzi

University of Colorado at Boulder
{Jinh,Awedh,Fabio}@Colorado.edu

Abstract. CirCUs is a satisfiability solver that works on a combination of And-Inverter-Graph, CNF clauses, and BDDs. It has been designed to work well with bounded model checking. It takes as inputs a Boolean circuit (e.g., the model unrolled k times) and an optional set of additional constraints (for instance, requesting that a solution correspond to a simple path) in the form of CNF clauses or BDDs. The algorithms in CirCUs take advantage of the mixed representation by applying powerful BDD-based implication algorithms, and decision heuristics that are *objective-driven*. CirCUs supports incremental SAT solving, early termination checks, and other analyses of the model that translate into SAT. Experimental results demonstrate CirCUs's efficiency.

1 Introduction

Efficient satisfiability (SAT) solvers [15, 18, 11, 6] have helped make Bounded Model Checking (BMC [2]) a widely used alternative to BDD-based model checking. The performance of BMC heavily depends on that of the SAT solver. At the same time, SAT-based model checking goes beyond the simple check for the existence of counterexamples, and involves a wide array of analyses such as early termination checks for invariants [13] and general LTL properties [1]. SAT solvers are also used in verification algorithms that combine BDDs and CNF [4], in the computation of concise proofs of satisfiability [12] and unsatisfiability [19], and in unbounded model checking [9, 10]. CirCUs is a SAT solver designed to be flexible enough to be used in all these tasks, while exploiting knowledge of the problem structure to provide better performance than generic SAT solvers in BMC.

CirCUs's input is a combination of And-Inverter-Graph (AIG [8]), CNF clauses, and BDDs. It combines the strengths of these different representations in a hybrid Boolean reasoning framework. The bounded model checker that uses CirCUs unrolls the model in the form of an AIG and applies optimizations like BDD sweeping and initial states propagation to it [8] to remove redundancy, which is a prime cause of inefficiency in SAT solvers. The constraints representing the property to be checked may be expressed as part of the AIG or as additional CNF clauses. Given these inputs, CirCUs transforms parts of the circuit into BDDs so as to apply powerful BDD-based implication algorithms [7]. It then looks for an assignment to the variables that satisfies all the outputs of the circuit and all additional constraints.

* This work was supported in part by SRC contract 2003-TJ-920.

R. Alur and D.A. Peled (Eds.): CAV 2004, LNCS 3114, pp. 519–522, 2004.

2 Objective-Driven Decisions

The speed of SAT solvers depends critically on their ability to choose good decision variables. In BMC, while the unrolled transition relation is always satisfiable, even after propagating the initial states, adding constraints on the target states makes most SAT instances unsatisfiable. CirCUs assumes that good decisions concentrate on proving that the *objective* cannot be satisfied, where the objective is an assertion on one of the outputs of the AIG. (If that assertion can be satisfied, then a counterexample is found.) Hence, decision variables are chosen from the cone of influence of the objective.

Even though the transition relation is satisfiable, a SAT solver will encounter conflicts in the attempt to derive consistent assignments for the inputs and outputs of subcircuits. These conflicts will produce *non-objective* clauses that express local satisfiability conditions for the transition relation and auxiliary objectives. Such clauses remain valid even when the objective changes as a result of further unrolling, and may prevent the search of fruitless parts of the solution space. When used as an incremental solver, CirCUs marks non-objective conflict clauses when they are created, and re-uses them for all successive runs and time frames. The clauses that depend on the objective, on the other hand, may be deleted by periodic clause deletion in CirCUs. Those that survive to the end of a run are "distilled" to bias the decision variable selection for the next run.

While this mechanism is not more powerful than the ones used in other generic incremental SAT solvers [17, 5] in identifying conflict clauses that remain valid from one run to the next, it either reduces the time required to detect what conflict clauses remain valid, or it removes the constraint that all conflict clauses must be kept. Compared to [14], CirCUs can identify more conflict clauses that remain valid.

3 Experimental Results

We have integrated CirCUs in VIS-2.1. To show the effectiveness of CirCUs, we compare the performance of four versions of BMC on the VIS benchmark suite [16]. All experiments have been performed on a 1.7 GHz Pentium IV with 1 GB of RAM running Linux. We have set the time out limit to 10000 s.

- Case A : VIS-2.0 [3, 16] BMC interfaced with Zchaff [11]
- Case B : VIS-2.1 BMC interfaced with Zchaff
- Case C : VIS-2.1 BMC interfaced with CirCUs
- Case D : VIS-2.1 BMC interfaced with Incremental CirCUs

We compare CPU times with scatter plots on logarithmic scale. The efficiency of optimizations like BDD sweeping and initial states propagation is shown in Fig. 1. In Case B, we run Zchaff on the CNF written from the optimized AIG. Since we use the same SAT solver for both case A and B, we can study the effects of removing redundancy. There are cases when redundancy gives an advantage in solving SAT. The score based decision heuristic may get a better initial decision order from redundant circuits. However, on average, redundancy elimination is very helpful.

In Fig. 2, we compare Zchaff's and (non-incremental) CirCUs's speed on the same BMC instances. CirCUs shows consistent improvement over Zchaff. CirCUs achieves

the larger speed-ups on the harder examples (up to 10x). Incremental vs. non-incremental SAT in CirCUs are compared in Fig. 3. Incremental SAT shows improvement especially for the harder cases. Because an incremental SAT run starts with additional clauses transferred from the previous run, in small examples it may incur significant overhead. We summarize the overall improvement of CirCUs over VIS-2.0 BMC in Fig. 4. Each scatterplot shows two lines: The main diagonal, and $y = \kappa \cdot x^{\eta}$, where κ and η are obtained by least-square fitting. Student's t test confirms that the improvement visible in each plot is statistically significant.

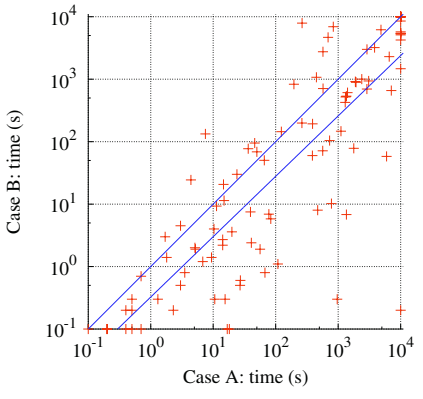

Fig. 1. Effects of redundancy removal

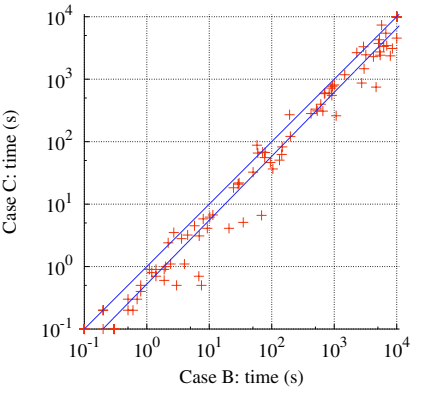

Fig. 2. CirCUs vs. Zchaff

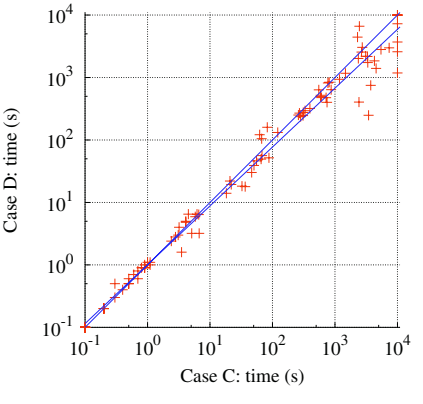

Fig. 3. Incremental vs. non-incremental

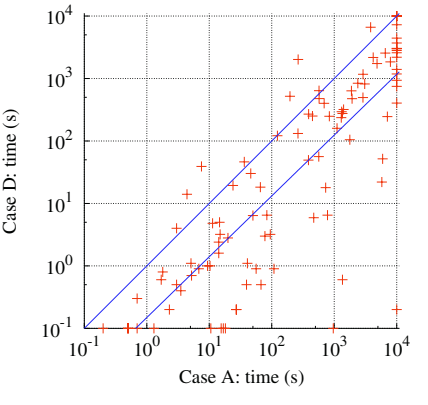

Fig. 4. Overall gains

Thanks to its performance and flexibility, CirCUs is suited for the modular development of abstraction refinement algorithms and other complex SAT-based applications.

References

1. M. Awedh and F. Somenzi. Proving more properties with bounded model checking. These proceedings.

2. A. Biere, A. Cimatti, E. Clarke, and Y. Zhu. Symbolic model checking without BDDs. In *Fifth International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'99)*, pages 193–207, Amsterdam, The Netherlands, Mar. 1999. LNCS 1579.
3. R. K. Brayton et al. VIS: A system for verification and synthesis. In T. Henzinger and R. Alur, editors, *Eighth Conference on Computer Aided Verification (CAV'96)*, pages 428–432. Springer-Verlag, Rutgers University, 1996. LNCS 1102.
4. G. Cabodi, S. Nocco, and S. Quer. Improving SAT-based bounded model checking by means of BDD-based approximate traversal. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 898–905, Munich, Germany, Mar. 2003.
5. N. Eén and N. Sörensson. Temporal induction by incremental SAT solving. *Electronic Notes in Theoretical Computer Science*, 89(4), 2003. First International Workshop on Bounded Model Checking. http://www.elsevier.nl/locate/entcs/.
6. E. Goldberg and Y. Novikov. BerkMin: A fast and robust SAT-solver. In *Proceedings of the Conference on Design, Automation and Test in Europe*, pages 142–149, Paris, France, Mar. 2002.
7. H. Jin and F. Somenzi. CirCUs: Speeding up circuit SAT with BDD-based implications. Submitted for publication, Apr. 2004.
8. A. Kuehlmann, M. K. Ganai, and V. Paruthi. Circuit-based Boolean reasoning. In *Proceedings of the Design Automation Conference*, pages 232–237, Las Vegas, NV, June 2001.
9. K. L. McMillan. Applying SAT methods in unbounded symbolic model checking. In E. Brinksma and K. G. Larsen, editors, *Fourteenth Conference on Computer Aided Verification (CAV'02)*, pages 250–264. Springer-Verlag, Berlin, July 2002. LNCS 2404.
10. K. L. McMillan. Interpolation and SAT-based model checking. In W. A. Hunt, Jr. and F. Somenzi, editors, *Fifteenth Conference on Computer Aided Verification (CAV'03)*, pages 1–13. Springer-Verlag, Berlin, July 2003. LNCS 2725.
11. M. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the Design Automation Conference*, pages 530–535, Las Vegas, NV, June 2001.
12. K. Ravi and F. Somenzi. Minimal assignments for bounded model checking. In *International Conference on Tools and Algorithms for Construction and Analysis of Systems (TACAS'04)*, pages 31–45, Barcelona, Spain, Apr. 2004. LNCS 2988.
13. M. Sheeran, S. Singh, and G. Stålmarck. Checking safety properties using induction and a SAT-solver. In W. A. Hunt, Jr. and S. D. Johnson, editors, *Formal Methods in Computer Aided Design*, pages 108–125. Springer-Verlag, Nov. 2000. LNCS 1954.
14. O. Shtrichman. Pruning techniques for the SAT-based bounded model checking problem. In *Correct Hardware Design and Verification Methods (CHARME 2001)*, pages 58–70, Livingston, Scotland, Sept. 2001. Springer. LNCS 2144.
15. J. P. M. Silva and K. A. Sakallah. Grasp—a new search algorithm for satisfiability. In *Proceedings of the International Conference on Computer-Aided Design*, pages 220–227, San Jose, CA, Nov. 1996.
16. URL: http://vlsi.colorado.edu/∼vis.
17. J. Whittemore, J. Kim, and K. Sakallah. SATIRE: A new incremental satisfiability engine. In *Proceedings of the Design Automation Conference*, pages 542–545, Las Vegas, NV, June 2001.
18. H. Zhang. SATO: An efficient propositional prover. In *Proceedings of the International Conference on Automated Deduction*, pages 272–275, July 1997. LNAI 1249.
19. L. Zhang and S. Malik. Validating SAT solvers using an independent resolution-based checker: Practical implementations and other applications. In *Design, Automation and Test in Europe (DATE'03)*, pages 880–885, Munich, Germany, Mar. 2003.